Facultad
de
Ciencias

Problemas de localización
Location problems

Trabajo Fin de Grado para acceder al
Grado en Matemáticas

Autor: Gustavo Bravo Viadero

Director: Francisco Santos Leal

Junio 2019

Índice general

Resumen

A la hora de establecer una actividad empresarial es importante saber cuál es la mejor ubicación para un negocio o servicio nuevo. Este es el llamado *problema de localización*. De entre las diversas técnicas que hay para este problema, en esta memoria estudiamos dos. En ambos casos se nos da como "input" un conjunto de puntos en el plano:

El *problema de Weber* consiste en encontrar un punto en el plano que minimice la suma de distancias a los puntos dados, a los que, además, se puede asignar distintos pesos. Este problema generaliza la mediana geométrica o *problema de Fermat* y por esta razón, a veces se le denomina problema de Fermat-Weber.

El *diagrama de Voronoi* es una descomposición del plano en regiones asociadas a los puntos u objetos dados. A cada punto se le asigna la región del espacio formada por los puntos que son más cercanos a él que a ninguno de los otros objetos.

Palabras clave: Localización, optimización geométrica, diagrama de Voronoi, problema de Weber

Abstract

When we want to start a business activity it is important to know what is the best place to open a new branch or service. This is known as *location problem*. Among the several techniques for this problem, here we study two of them. In both of them we are given as "input" a set of points in the plane:

The *Weber problem* consists in finding the point in the plane that minimizes the sum of distances to the given points to which, moreover, we can assign different weights. This problem generalizes the geometric median or *Fermat problem* and, for this reason, it is sometimes called Fermat-Weber problem.

The *Voronoi diagram* is a decomposition of the plane in regions associated to the given points or objects. To each point we assign the region of space formed by the points that are closer to it than to any other of the objects.

Keywords: Location, geometric optimization, Voronoi diagram, Weber problem

Capítulo 1

Introducción

La localización de una buena ubicación en donde vamos a desarrollar una actividad empresarial es una decisión de gran importancia, aunque con demasiada frecuencia no se tiene en cuenta. A la hora de escoger la localización de un negocio, hay que elegir entre varios criterios, como pueden ser cuestiones de costo, rentabilidad, tiempos de respuesta, cercanía a determinados lugares, etc. Todos ellos dependerán en función a las características de la empresa. Por otro lado, en los casos en los que se tienen en cuenta pocos criterios puede ser perjudicial a la hora de ubicar nuestro negocio.

Hay dos razones que justifican la gran importancia de la decisión de la localización de nuestro negocio. En primer lugar, hay que realizar una gran inversión debido a que las instalaciones son por lo general muy costosas. una vez construidas, las inversiones realizadas no se podrían recuperar sin sufrir grandes pérdidas. (En algunos casos se puede optar por instalaciones provisionales poco costosas o alquileres, reduciendo el riesgo de pérdidas económicas). En segundo lugar, una buena elección de la ubicación favorecerá al desarrollo del negocio en todas sus áreas.

A la hora de tomar la decisión sobre la localización de nuestro negocio hay que tener en cuenta varios factores. A continuación, detallamos algunas de ellas:

- La ubicación (encontrar un costo similar en todas partes) y localización de materiales.
- Factores de demanda.
  - Cercanía de los potenciales clientes.
  - Localización de competidores que a su vez determinan parcialmente, la cantidad de demanda y la elasticidad cruzada de demanda en diferentes lugares.
  - El significado de proximidad,(tipo de servicio y rapidez de servicio).
  - Relación entre el personal de contacto y de venta.
  - La competencia industrial en la localización y en el precio.

- Coste del negocio.
  - Coste de la tierra, la cual incluye el alquiler de tierras, impuestos, disponibilidad de capital, tasas y costes de gasolina.
  - Coste de mano de obra.
  - Coste de material y equipamiento.
  - Coste de transportes, los que se incluye la topografía del lugar.
  - Factores puramente personales

También puede haber factores menos directos pero igualmente importantes como pueden ser:

- El gobierno presente en la región.
- Los mercados.
- Precio de construcción.

Para modelizar y cuantificar este tipo de factorse vamos a estudiar dos problemas matemáticos de localización. Estos problemas son:

- El problema de Weber, el cual trata de encontrar el punto que minimiza la suma de distancias a n puntos en el plano. A menudo este problema es conocido como el problema de Fermat-Weber, figura 1.1. Dentro de este apartado veremos el problema de los tres puntos (el más sencillo, sección 2.2) con su resolución, el problema dual de Weber con un enfoque diferente para encontrar la ubicación de la mínima suma. El Análogo Mecánico el cual nos da una idea de como localizar la mejor ubicación a a través del armazón de Varignon. EL algoritmo de Weiszfeld con el cual resolveremos unos problemas ya planteados.

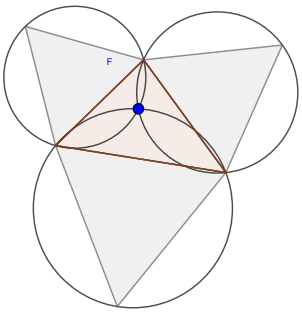

Figura 1.1: Punto de Fermat

- El diagrama de Voronoi, figura1.2, el cual es una construcción geométrica que, dado un conjunto S de n puntos, divide el plano en n regiones asociadas a los n puntos.. En este apartado tratamos de demostrar una serie de propiedades que cumplen los diagramas de Voronoi, los cuales nos van a servir para realizar y comprender posteriormente los problemas realizados. También veremos algunas aplicaciones de los diagramas de Voronoi. Por último trataremos de explicar qué es el diagrama de Voronoi de puntos más lejanos y algunas de sus propiedades.

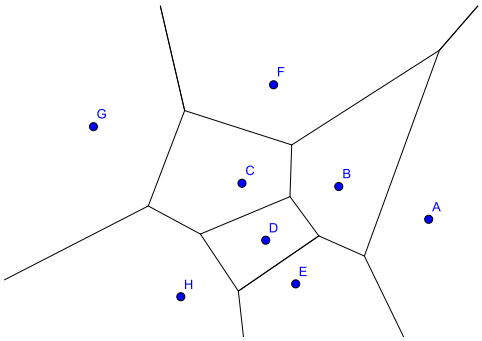

Figura 1.2: Diagrama de Voronoi

1.1. Definiciones generales de convexidad

La idea de conjunto convexo es que es aquél conjunto que contiene cualquier segmento que une dos puntos del conjunto. Es decir:

Definición 1.1.1.

Un conjunto $C \subset \mathbb{R}^d$ es *convexo* si para todo $\mathbf{p}, \mathbf{q} \in C$ el segmento $[\mathbf{pq}] \subset C$.

Sea C un subconjunto convexo de $\mathbb{R}$. Sea $f : C \to \mathbb{R}$ una función definida en un convexo C de $\mathbb{R}^d$. Diremos que dicha *función* es *convexa* en C si todo segmento que une dos puntos de la gráfica queda por encima de la gráfica. Es decir, si para cualquier par de puntos $\mathbf{p}, \mathbf{q} \in C$ y para cualquier $\lambda \in [0, 1]$ $f((1-\lambda)\mathbf{p} + \lambda\mathbf{q}) \leq (1-\lambda)f(\mathbf{p}) + \lambda f(\mathbf{p})$.

Se dice que la función es estrictamente convexa en C si $f((1-\lambda)\mathbf{p} + \lambda\mathbf{q}) < (1-\lambda)f(\mathbf{p}) + \lambda f(\mathbf{q})$ siempre que $\lambda \in (0, 1)$.

Se llama *envolvente convexa* de un conjunto dado de C al menor conjunto convexo que contiene a C. Es decir,

$$\text{conv}(C) = \{\lambda_1\mathbf{p}_1 + \cdots + \lambda_k\mathbf{p}_k \mid \mathbf{p}_1, \ldots, \mathbf{p}_k \in C, \lambda_i \geq 0 \forall i, \sum \lambda_i = 1\}.$$

El concepto de "menor convexo que" tiene sentido gracias a:

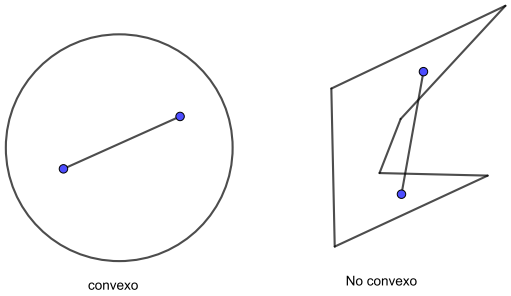

Figura 1.3: Conjunto convexo

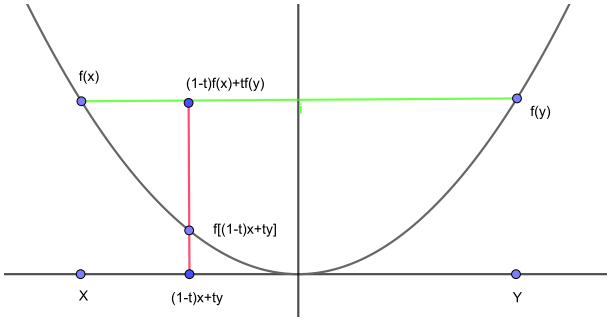

Figura 1.4: Función convexa

Teorema 1.1.2. *Toda intersección de convexos es un convexo.*

Demostración. Sea $\{C_i\}_{i\in I}$ una familia de conjuntos convexos en R^n. Queremos ver que $\cap_{i\in I}C_i$ es un conjunto convexo en R^n.

Es decir, sean $\mathbf{p}, \mathbf{q} \in \cap_{i\in I}C_i$ y queremos ver que $[\mathbf{pq}] \subset \cap_{i\in I}C_i$. Como $\mathbf{p}, \mathbf{q} \in \cap_{i\in I}C_i$ entonces $\mathbf{p}, \mathbf{q} \in C_i \ \forall i \Rightarrow [\mathbf{pq}] \subset C_i \ \forall i$. Por lo tanto $[\mathbf{pq}] \subset \cap_{i\in I}C_i$. □

Capítulo 2

Problema de Weber

2.1. Definición problema de Weber

Se llama problema de Weber al problema de encontrar el punto que minimiza la suma de distancias euclidianas ponderadas a n puntos fijos del plano. Es decir, tenemos n puntos $\mathbf{p}_i = (a_i, b_i) \in \mathbb{R}^2$ con $i = \{1, \ldots, n\}$ y tenemos pesos $w_1, \ldots, w_n \in (0, \infty)$ asociados a ellos, y queremos encontrar el punto (x^*, y^*) que minimiza la función

$$\min_{x,y}\{W(x, y) = \sum_{i=1}^{n} w_i d_i(x, y)\} \tag{2.1}$$

donde $d_i(x, y) = \sqrt{(x - a_i)^2 + (y - b_i)^2}$ es la distancia euclidea entre (x, y) y (a_i, b_i).

Un escenario simple donde el problema aparece de manera natural para el problema es que queremos colocar un almacén y que los pesos w_i son los costos por unidad de distancia de envío de los requisitos a los clientes (y/o desde los proveedores) ubicados en los puntos fijos (a_i, b_i); (x^*, y^*) es entonces la ubicación del almacén que minimiza el costo total de transporte.

El problema de Weber es una generalización del problema de Fermat, que es el caso en el que todos los pesos w_i son iguales a 1, es decir, queremos minimizar la suma de distancias sin ponderar. En este caso también se puede entender (x^*, y^*) como una generalización bidimensional de la mediana de n valores ponderados, por lo cual al problema de Fermat, y a su generalización a cualquier dimensión, a veces se le llama mediana geométrica o mediana espacial.

El problema de Weber generaliza la mediana geométrica, que asume que los costos de transporte por unidad de distancia son los mismos para todos los puntos de destino, y el problema de calcular el punto de Fermat, que es la mediana geométrica de tres puntos. Este último fue formulado por el famoso matemático francés Pierre de Fermat en 1640, y puede verse como el comienzo de la teoría de la localización. Por esta razón, al problema de Weber se le denomina a veces como problema Fermat-Weber. Torricelli encontró una solución geométrica

al problema de Fermat alrededor de 1645, pero todavía no tenía una solución numérica. Kuhn y Kuenne encontraron una solución iterativa para el problema general de Fermat en 1962, solución que se aplica al caso de más de tres puntos. En 1972 Tellier dio una solución numérica al problema del triángulo de Fermat.

2.2. Problema de tres puntos

Lema 2.2.1. *Sean A, B y C tres puntos del plano tales que el triángulo ABC no tiene ningún ángulo mayor que* 120^o*. Entonces un punto O es el punto de Fermat de A, B y C si y solo si los tres ángulos AOB, BOC y COA son* 120^o

Demostración. Al no tener el triángulo ABC ningún ángulo mayor de 120^o, existe un punto O tal que los tres ángulos AOB, BOC y COA son de 120^o. Lo que tenemos que demostrar es que si X es cualquier otro punto del plano entonces

$$|\overrightarrow{OA}| + |\overrightarrow{OB}| + |\overrightarrow{OC}| < |\overrightarrow{XA}| + |\overrightarrow{XB}| + |\overrightarrow{XC}|.$$

Sean $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ los vectores unitarios de O a lo largo de $\overrightarrow{OA}$, $\overrightarrow{OB}$, $\overrightarrow{OC}$, de modo que

$$|\overrightarrow{OA}| = \overrightarrow{OA} \cdot \mathbf{i} = \overrightarrow{OX} \cdot \mathbf{i} + \overrightarrow{XA} \cdot \mathbf{i}.$$

Haciendo lo mismo para $\mathbf{b}$ y $\mathbf{c}$ y sumando las ecuaciones obtenemos que

$$\begin{aligned} |\overrightarrow{OA}| + |\overrightarrow{OB}| + |\overrightarrow{OC}| &= \overrightarrow{XA} \cdot \mathbf{i} + \overrightarrow{XB} \cdot \mathbf{j} + \overrightarrow{XC} \cdot \mathbf{k} + \overrightarrow{OX} \cdot (\mathbf{i} + \mathbf{j} + \mathbf{k}) \\ &= \overrightarrow{XA} \cdot \mathbf{i} + \overrightarrow{XB} \cdot \mathbf{j} + \overrightarrow{XC} \cdot \mathbf{k} \\ &\leq |\overrightarrow{XA}| + |\overrightarrow{XB}| + |\overrightarrow{XC}|, \end{aligned}$$

donde la igualdad sale de que al ser los tres ángulos 120^o se tiene que $\mathbf{i}+\mathbf{j}+\mathbf{k} = 0$ y la desigualdad es estricta porque para que hubiera igualdad sería necesario que

$$\overrightarrow{XA} \parallel \overrightarrow{OA}, \quad \overrightarrow{XB} \parallel \overrightarrow{OB}, \quad \overrightarrow{XC} \parallel \overrightarrow{OB},$$

lo cual solo pasa si $X = O$.

□

La construcción con los triángulos equiláteros es la manera de encontrarlo gráficamente, basada en el *teorema del arco capaz*: "dados dos puntos A y B del plano, los puntos X con la propiedad de que el ángulo AXB es una cierta constante α están en el arco de circunferencia que pasa por AB y cuyo centro O satisface que AOB es 2α". Véase Figura 2.1.

Para encontrar el punto de Fermat geométricamente, unimos los tres puntos por líneas para formar un triángulo. Luego construimos triángulos equiláteros en los tres lados, con los vértices apuntando hacia fuera. Los tres círculos circunscritos a los triángulos equiláteros se intersecan en el punto de Fermat. Véase Figura 2.2.

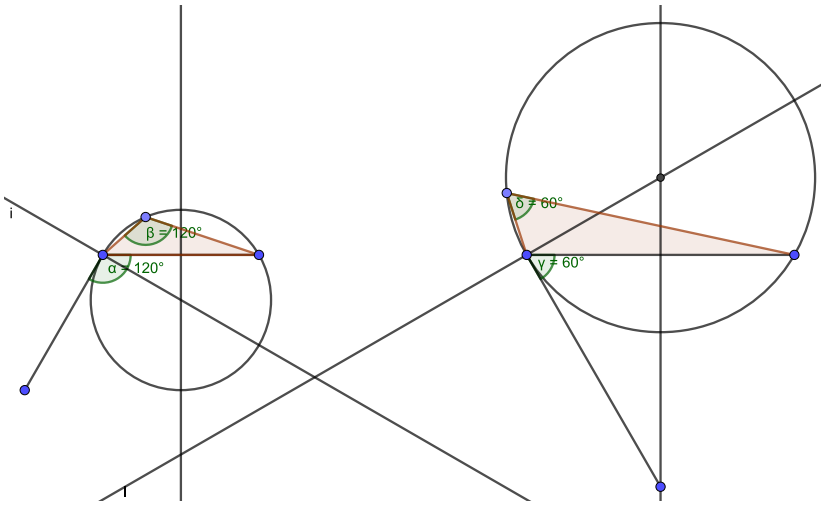

Figura 2.1: Arco capaz

2.3. Problema dual

Un enfoque diferente para encontrar la ubicación de la mínima suma o punto mediano espacial (x^*, y^*) es resolver un problema dual. Consideramos el problema de programación cuadrática

$$\max_{U,V}\{D(U,V) = -\sum_{i=1}^{n}(a_i u_i + b_i v_i)\} \tag{2.2}$$

sujeto a:

$$\sum_{i=1}^{n} u_i = 0 \tag{2.3}$$

$$\sum_{i=1}^{n} v_i = 0 \tag{2.4}$$

$$\sqrt{u_i^2 + v_i^2} \leq w_i \tag{2.5}$$

El valor máximo de D en (2.2) es igual al valor mínimo de W en (2.1). Los vectores $(u_i, v_i)^T$ apuntan desde los puntos fijos (a_i, b_i) hasta el punto de suma mínima (x^*, y^*), así que cualquiera de estos dos vectores puede ser utilizado para resolver su localización.

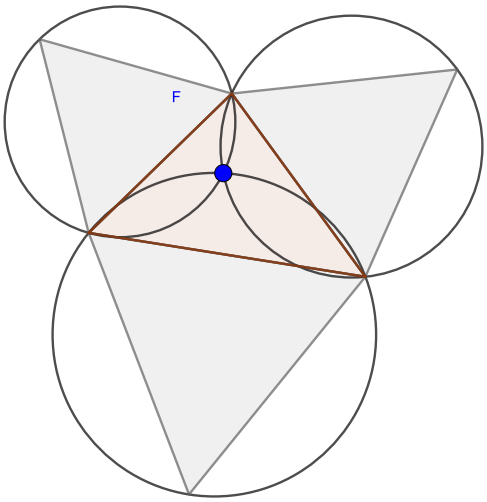

Figura 2.2: Punto de Fermat

2.4. Algoritmo de Weiszfeld

La técnica más simple y más comúnmente utilizada para resolver el problema de Weber se llama el procedimiento de Weiszfeld. Es un método iterativo que (habitualmente) converge a la solución óptima.

Si diferenciamos la expresión (2.1) e igualamos las derivadas parciales a cero para obtener las condiciones de primer orden para la optimalidad, tenemos:

$$\begin{aligned} \frac{\partial W(x,y)}{\partial x} &= \sum_{i=1}^{n} \frac{w_i(x-a_i)}{d_i(x,y)} = 0 \\ \frac{\partial W(x,y)}{\partial y} &= \sum_{i=1}^{n} \frac{w_i(y-a_i)}{d_i(x,y)} = 0 \end{aligned} \tag{2.6}$$

Se tiene que:

Lema 2.4.1. *$W(x,y)$ es convexa.*

Demostración. La suma de varias funciones convexas es siempre convexa, así que basta con demostrar que la función $\mathbf{x} \mapsto \text{dist}(\mathbf{p},\mathbf{x})$ es convexa, donde $\mathbf{p} \in \mathbb{R}^2$ es un punto cualquiera. Haciendo un esquema del triángulo $\mathbf{pxy}$ y de un punto $\mathbf{q} = \lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ en el segmento $\mathbf{xy}$ sale que la convexidad es consecuencia de la desigualdad triangular. □

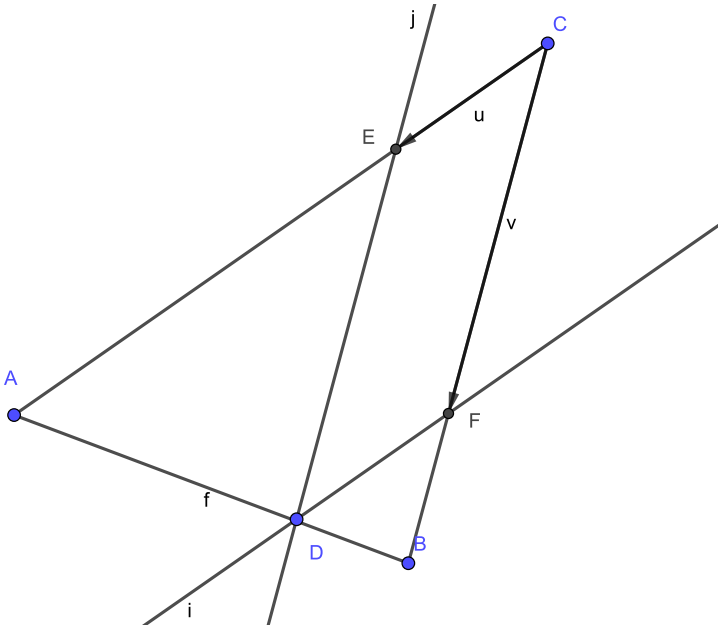

Figura 2.3: La función W es convexa

Por tanto, (2.6) define un único mínimo. Sin embargo, es inmediatamente evidente que estas derivadas no existen cuando (x, y) coincide con alguno de los puntos (a_i, b_i), porque entonces $d_i(x, y) = 0$. Las ecuaciones no pueden, en general, ser resueltas explícitamente para (x, y) si $n > 3$. Por esta razón decimos que el método no siempre converge: si en alguna iteración caemos en uno de los puntos dados el método no puede continuar. Veremos un ejemplo de este fenómeno más adelante.

Podemos extraer x de la primera ecuación en (2.6) si ignoramos su presencia en $d_i(x, y)$ y hacemos lo mismo para y de la segunda ecuación en (2.6). El resultado puede ser formulado como un procedimiento iterativo si consideramos la (x, y) despejada como una nueva iteración $(k + 1)$ y la (x, y) de d_i como la iteración antigua (k). De manera más específica:

$$(x^{(k+1)}, y^{(k+1)}) = \left(\frac{\sum_{i=1}^{n} w_i a_i / d_i(x^k, y^k)}{\sum_{i=1}^{n} w_i / d_i(x^k, y^k)}, \frac{\sum_{i=1}^{n} w_i b_i / d_i(x^k, y^k)}{\sum_{i=1}^{n} w_i / d_i(x^k, y^k)} \right) \tag{2.7}$$

En otras palabras, una vez que tenemos un punto (x^k, y^k), podemos obtener el siguiente, y esperamos que sea mejor, sustituyendo en el lado derecho de (2.7). El truco de despejar parcialmente las variables de la solución con el fin de obtener un método de solución iterativo es bien conocido en el análisis numérico

y pertenece a la clase de procedimientos conocidos como *métodos de iteración de un punto*. la expresión *un punto* proviene de que sólo el punto actual se usa para determinar el siguiente.

Hay que tener en cuenta casos especiales en el que el método de Weiszfeld no converge, esto normalmente ocurre debido a que la solución coincide casi por completo con uno de los puntos de fijos. En algunos casos esto puede ser el resultado de una convergencia extremadamente lenta del proceso de Weiszfeld. Cabe señalar que Ostresh mostró que este problema se puede resolver en la mayoría de los casos identificando que el punto de demanda donde el método está estancado no es la solución óptima, alejándose de ella y continuando con las iteraciones de Weiszfeld. Obsérvese que en algunos casos sí puede ocurrir que uno de los puntos de demanda a_i sea a la vez el ópimo. Varios autores mostraron que esto ocurre si y solo si su peso es mayor que la norma de la suma de los vectores $w_j \mathbf{p}_i \vec{\mathbf{p}}_j$ para todos los otros puntos de demanda. Es decir:

$$\left[\left(\sum_{i=1}^{n} \frac{w_i(x_i - x_k)}{d_i(x_k, y_k)} \right)^2 + \left(\sum_{i=1}^{n} \frac{w_i(y_i - y_k)}{d_i(x_k, y_k)} \right)^2 \right]^{1/2} \leq w_k \tag{2.8}$$

Una explicación es mediante el mecanismo de Varignon que introduccimos en la siguiente sección: Si al colocar (x^*, y^*) en uno de los puntos de demanda $\mathbf{p}_i$, el peso w_i hacia abajo es mayor que la resultante de las demás fuerzas, el mecanismo no se va a mover.

2.5. El análogo mecánico

Varignon propuso un dispositivo mecánico que simula el problema de Weber y que arroja información valiosa sobre el problema. La figura 2.4 muestra un diagrama del dispositivo, que se denomina *armazón de Varignon*. Una tabla se perfora con n agujeros correspondientes a las coordenadas de los n puntos fijos; n cuerdas se unen en un nudo en un extremo, los cabos sueltos se pasan uno por cada agujero, y se unen a pesos físicos debajo del tablero que tienen las mismas magnitudes que las constantes w_i. Si el dispositivo no estuviera sujeto a los males del mundo físico y no hubiera fricción, si las cuerdas fueran infinitamente delgadas, los agujeros infinitamente pequeños y así sucesivamente, entonces la posición final del nudo estaría en el punto (x^*, y^*).

¿Por qué es el punto óptimo en el nudo? Consideramos la componente de la fuerza en la dirección x, ejercida por un solo peso, cuando el nudo está en coordenadas (x, y). Esta componente de fuerza en la dirección x será $w_i(x - a_i)/d_i(x, y)$ e irá hacia la izquierda (tendrá un signo negativo) si y solo si $a_i > x$. Es evidente que la suma de dichas componentes de todos los pesos es igual a la primera expresión de la derivada de Weiszfeld. Del mismo modo, la suma de los componentes en la dirección negativa y es la segunda expresión de la derivada de Weiszfeld. El vector de fuerza resultante ejercido sobre el nudo por todas las cuerdas es por lo tanto cero en el mismo punto (x^*, y^*) donde la condición de

optimalidad de las derivadas de Weiszfeld se cumple. (Hay que tener en cuenta que si cualquiera de los agujeros en el armazón de Varignon se mueve hacia afuera en la línea de su cuerda con el nudo, el punto óptimo no se ve afectado, esto es análogo a la propiedad de la mediana simple que en los puntos que no coinciden con la mediana se pueden estirar o intercambiar en cualquier lado a la vez sin afectar la mediana.)

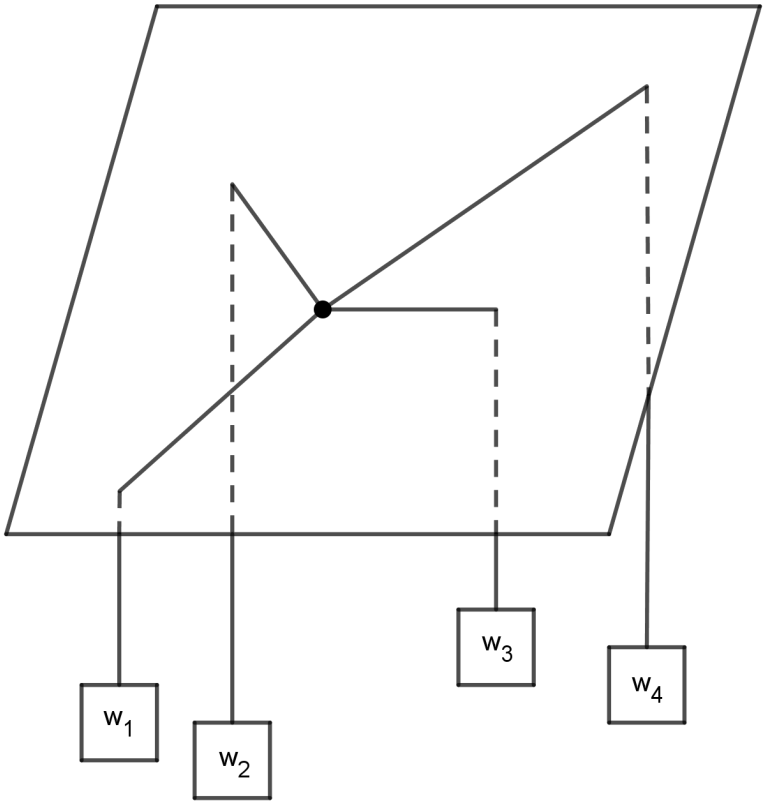

Figura 2.4: EL armazón de Varignon

2.6. Otros métodos iterativos

Austin (1959) obtuvo las ecuaciones de iteración, pero las utilizó de una manera diferente. Señaló que para cualquier punto de partida, mostraban una

solución de centroide con pesos inversamente proporcionales a las distancias. Seymour (1970) comparó este método computacionalmente con el algoritmo de Weiszfeld. Sin embargo, se puede demostrar fácilmente que este es un método simple de gradiente con un tamaño de paso fijo, mientras que el procedimiento de Weilszfeld también es de pendiente descendente (Cooper y Katz, 1981), pero tiene un tamaño de paso variable de $\sum \frac{wi}{d_i(x^k),y^k}$.

Se sabe que la convergencia del procedimiento Weiszfeld es lenta cuando (x, y) está cerca de alguno de los puntos de demanda. Usando técnicas estándar de minimización no lineal, se podría eliminar el problema de las derivadas en los puntos fijos utilizando una aproximación hiperbólica. Drezner (1986) propuso una interpretación de esta idea suponiendo que los puntos de demanda teienen un área. Entonces la distancia promedio entre (x, y) y un "punto" de demanda se asemeja a $d_i^H = \sqrt{(x - a_i)^2 + (y - b_i)^2 + \varepsilon}$ siendo ε proporcional al área del punto de demanda. Desafortunadamente, usar esta aproximación puede hacer que la función objetivo se acerque al óptimo, pero que el punto encontrado no esté cerca del verdadero punto óptimo, si la función de costo es muy plana.

Vergin y Rodgers (1967) utilizaron métodos de degradado y Love (1969) aplicó la programación convexa a un problema en tres dimensiones.

Capítulo 3

Diagramas de Voronoi

El diagrama de Voronoi es una construcción geométrica que asigna a cada punto una región, de forma que todo lo que contiene esa región está más cerca de este punto que de cualquier otro. Es muy útil para resolver problemas de proximidad asociados a un conjunto de puntos dado.

3.1. Definición

Definición 3.1.1. Dado un conjunto $S = \{p_1, \ldots, p_n\}$ de puntos en el plano Euclídeo, llamamos *región de Voronoi* del punto p_i con respecto al conjunto S a la formada por los puntos del plano que distan de p_i menos que de cualquier otro punto de S. Es decir:

$$V_S(p_i) = \{x \in \mathbb{R}^2 : d(x, p_i) < d(x, p_j)\ \forall p_j \in S \setminus \{p_i\}\}.$$

Cada región de Voronoi es el interior de una región poligonal definida como intersección de semiplanos. Si llamamos $H_{ij} := \{x \in \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j)\}$ tenemos que $V_S(p_i)$ es el interior de $\cap_{j \in \{1,..,n\} \setminus \{i\}} H_{ij}$. Podemos por tanto hablar de los *vértices* y *aristas* de cada región de Voronoi.

Definición 3.1.2. Dado un conjunto $S = \{p_1, \ldots, p_n\}$ de puntos en el plano Euclídeo, llamamos *diagrama de Voronoi* a la partición del plano formada por las regiones de Voronoi, sus vértices y sus aristas. Lo denotamos como $\mathrm{Vor}(S)$.

3.2. Historia

El Diagrama de Voronoi lleva el nombre del matemático ucraniano residente en el Imperio ruso Georgy Fedosievych Voronoyi definió, que estudió el caso n-dimensional general en 1908. También se le conoce como teselación de Voronoi , descomposición de Voronoi , o partición de Voronoi. Los diagramas de Voronoi tienen aplicaciones prácticas y teóricas en una gran cantidad de campos, principalmente en ciencia y tecnología. El uso informal de los diagramas de

Voronoi se remonta a Descartes en 1644. Peter Gustav Lejeune Dirichlet utilizó diagramas de Voronoi bidimensionales y tridimensionales en su estudio de formas cuadráticas en 1850. El médico británico John Snow usó un diagrama de Voronoi en 1854 para ilustrar cómo la mayoría de las personas que murieron en el brote de cólera del Soho de Londres vivían más cerca de la bomba de Broad Street que de cualquier otra bomba de agua, y así concluir que dicha bomba estaba infectada y era el origen del brote.

3.3. Un algoritmo para calcular diagramas de Voronoi en el plano

Para construir un diagrama de Voronoi debemos construir todos sus polígonos. Puesto que cada polígono Voronoi es la intersección de $n-1$ semiplanos, se puede construir en el tiempo $O(n \log n)$, dando por resultado un tiempo global $O(n^2 \log n)$. Sin embargo, como veremos a continuación, es posible construir el diagrama entero en tiempo óptimo $O(n \log n)$. Eso significa que construir el diagrama es igual de difícil que construir uno solo de sus polígonos.

De hecho, a pesar de su aparente complejidad, el diagrama de Voronoi es muy adecuado para atacarlo por un método de tipo divide-y-vencerás. El éxito del método depende de varias propiedades estructurales del diagrama que nos permiten fusionar subproblemas.

Procedimiento diagrama Voronoi:

1. Partir S en dos subconjuntos S_1 y S_2 de tamaños aproximadamente iguales.
2. Construir $\text{Vor}(S_1)$ y $\text{Vor}(S_2)$ recursivamente.
3. Combinar $\text{Vor}(S_1)$ y $\text{Vor}(S_2)$ para obtener $\text{Vor}(S)$.

Obviamente el paso 3 es el más complicado, y más adelante hablaremos de él.

Para analizar la complejidad del procedimiento, suponemos ahora que la Etapa 1 puede llevarse a cabo en el tiempo $O(n)$. Si denotamos como $T(n)$ el tiempo total de ejecución del algoritmo, entonces el Paso 2 se completa en tiempo aproximadamente $2T(n/2)$. Si somos capaces de fusionar los dos diagramas $\text{Vor}(S_1)$ y $\text{Vor}(S_2)$ (paso 3) en tiempo $O(n)$, entonces tendremos una fórmula recursiva

$$T(n) \leq O(n) + 2T(n/2) + O(n),$$

que implica que $T(n) \in O(n \log n)$.

Definición 3.3.1. Dada una partición S_1, S_2 de S, denotamos como $\sigma(S_1, S_2)$ el conjunto de aristas de Voronoi de $\text{Vor}(S)$ que son compartidos por un par de polígonos $V(i)$ y $V(j)$ con $p_i \epsilon S_1$ y $p_j \epsilon S_2$.

Teorema 3.3.2. *$\sigma(S_1, S_2)$ es el conjunto de borde de un subgrafo de* $\mathrm{Vor}(S)$ *con las propiedades:*
(i) $\sigma(S_1, S_2)$ consta de ciclos de aristas disjuntos. Si una cadena tiene sólo un borde, ésta es una línea recta; De lo contrario sus dos bordes extremos son rayos semi-infinitos
(ii) Si S_1 y S_2 están separados linealmente, entonces, $\sigma(S_1, S_2)$ consiste en una única cadena monótona.

Demostración. (i) Si en $\mathrm{Vor}(S)$ imaginamos pintar cada uno de los polígonos $V(i) : p_i \epsilon S_1$ en rojo y cada uno de los polígonos $V(j) : p_j \epsilon S_2$ en verde, tenemos $\mathrm{Vor}(S)$ como un mapa de dos colores. Entonces los límites entre polígonos de diferentes colores son ciclos de borde disjuntos. Cada componente de $\sigma(S_1, S_2)$ divide el plano en dos partes. Así, una cadena consiste en una sola línea recta o tiene rayos como bordes inicial y final.
(ii)Supongamos que S_1 y S_2 están separadas por una línea vertical m, y que C sea una componente de $\sigma(S_1, S_2)$. Partiendo de un punto q de un borde de C, comenzamos a recorrer C en la dirección que decrece y y continuar hasta llegar a un vértice de Voronoi v_1 donde C gira hacia arriba. El borde $(\overline{v_1 v_2})$ es la bisectriz de $(\overline{p_1 p_2})$, y $(\overline{v_1 v_3})$ es la bisectriz de $(\overline{p_1 p_3})$.

Supongamos ahora que $\sigma(S_1, S_2)$ contiene al menos dos cadenas, C_1 y C_2. Una línea horizontal l interseca a cada C_1 y C_2 en un solo punto. Supongamos también que la intersección con C_1 está a la izquierda de C_2. Entonces hay tres puntos de S : p_1, p_2 y p_3, con $x(p_1) < x(p_2) < x(p_3)$, de los cuales p_1 y p_3 pertenecen al mismo conjunto de la partición S_1, S_2, contradiciendo la Hipótesis de separación vertical de estos dos conjuntos. Por lo tanto, $\sigma(S_1, S_2)$ consiste en una cadena monótona. Si la línea de separación m se elige vertical, podemos decir inequívocamente que σ corta el plano en una porción izquierda π_L y una porción derecha π_R. □

Teorema 3.3.3. *Si S_1 y S_2 están linealmente separados por una línea vertical con S_1 a la izquierda de S_2, entonces el diagrama Voronoi* $\mathrm{Vor}(S)$ *es la unión de* $\mathrm{Vor}(S_1) \cap \pi_L$ *y* $\mathrm{Vor}(S_2) \cap \pi_R$*, donde π_L (respectivamente π_R) es la parte de* $\mathrm{Vor}(S_1)$ *(resp. de* $\mathrm{Vor}(S_2)$*) que queda a la izquierda (resp. a la derecha) de la cadena $\sigma(S_1, S_2)$.*

Para una ilustración de este enunciado véase la Figura 3.1, donde los diagramas $\mathrm{Vor}(S_1)$, $\mathrm{Vor}(S_2)$ y la poligonal σ se muestran superpuestos.

Demostración. Todos los puntos de S a la derecha de $\sigma(S_1, S_2)$ son los puntos de S_2. Entonces todas los aristas de $\mathrm{Vor}(S)$ a la derecha de $\sigma(S_1, S_2)$ separan los polígonos $V(i)$ y $V(j)$, donde tanto p_i como p_j están en S_2. Esto implica que cada borde de $\mathrm{Vor}(S)$ en π_R coincide o es una porción de un borde de $\mathrm{Vor}(S_2)$. Lo mismo ocurre para π_L. □

El teorema 3.3.3 responde a nuestra pregunta original sobre cómo relacionar $\mathrm{Vor}(S_1)$ y $\mathrm{Vor}(S_2)$ con $\mathrm{Vor}(S)$. De hecho, proporciona un método para fusionar $\mathrm{Vor}(S_1)$ y $\mathrm{Vor}(S_2)$ cuando S_1 y S_2 están separados linealmente. El algoritmo revisado queda como sigue:

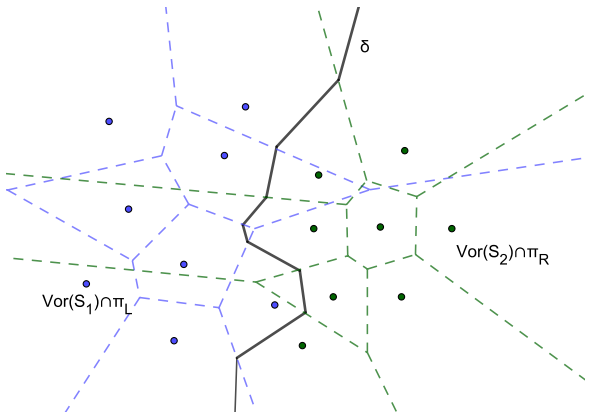

Figura 3.1: Los diagramas Vor(S_1) (azul) y Vor(S_2) (verde), junto a la poligonal σ (negro)

Procedimiento diagrama Voronoi (más detallado):

1. Partir de S en dos subconjuntos S_1 y S_2, de aproximadamente tamaños iguales, por la mediana de la coordenada x.

2. Construir Vor(S_1) y Vor(S_2) recursivamente.

3.1. Construir la cadena poligonal $\sigma(S_1, S_2)$, separando S_1 y S_2

3.2. Deshacer todas los aristas de Vor(S_2) que se encuentran a la izquierda de σ y todos los aristas de Vor(S_1) que se encuentran a la derecha de σ.

3.3. Lo que queda de Vor(S_1) y Vor(S_2), junto con la polygonal $\sigma(S_1, S_2)$ es Vor(S), el diagrama voronoi de todo el conjunto.

Claramente, el éxito de este procedimiento depende de la rapidez con que seamos capaces de construir $\sigma(S_1, S_2)$, (paso 3.1) ya que los pasos 3.2 y 3.3 no plantean dificultades. Desde un punto de vista de rendimiento, la partición inicial de S de acuerdo con la mediana de las coordenadas x puede realizarse en el tiempo $O(n)$ por algoritmos estándar de búsqueda de la mediana. Además, el último paso se puede llevar a cabo en el tiempo $O(|S_1| + |S_2|) = O(n)$.
Ahora tendríamos que describir una manera eficiente de construir la cadena divisoria σ, pero como nuestro trabajo está enfocado en la localización no profundizaremos en este tema: los detalles se puedens, por ejemplo, encontrar explicado en el libro [12, pp. 209–214].

3.4. Propiedades de los diagramas de Voronoi

3.4.1. Relación con circuncentros

Lema 3.4.1. *Sean $p_1, p_2, p_3 \in S$ tres puntos cuyas regiones de Voronoi comparten un vértice v. Entonces la circunferencia C que pasa por p_1, p_2 y p_3 tiene su centro en v y no contiene ningún punto de S en su interior.*

Demostración. Como v está en la clausura de las tres regiones de Voronoi, su distancia a p_1, p_2 y p_3 es la misma; por tanto v es el centro de la circunferencia C. Si dentro de C hay algún otro punto, digamos p_4, p_4 está más próximo a v que a cualquiera de p_1, p_2 o p_3, en cuyo caso v debe estar en $V(4)$ y no en la clausura de $V(1)$, $V(2)$ o $V(3)$, por la definición de un polígono de Voronoi. Pero esto es una contradicción ya que v es común a $V(1)$, $V(2)$ y $V(3)$. Ver Figure 3.2. □

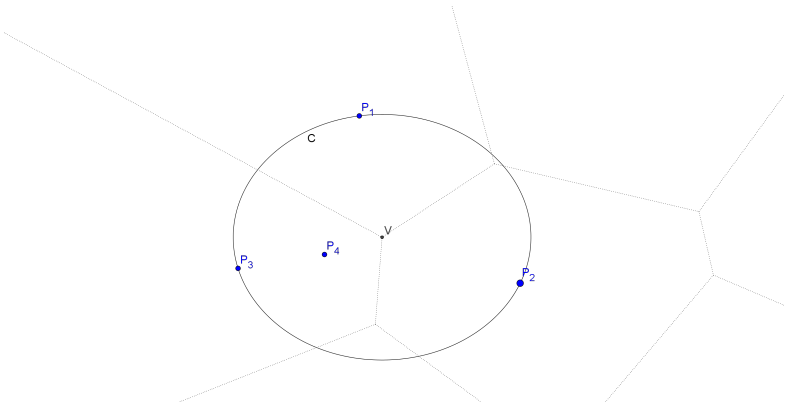

Figura 3.2: El círculo no contiene otros puntos de S.

3.4.2. El punto que maximiza la distancia a S es un vértice de Voronoi

Supongamos que queremos colocar algo lejos de ciertos puntos ya dados. Esto puede ser por (1) queremos colocar algo necesario pero desagradable lejos de los núcleos urbanos. (2) queremos colocar un servicio en un sitio donde haya escasez (abrir un negocio lejos de otros negocios parecidos, o poner un centro de salud/colegio, etc en una zona donde hace falta, pero lejos de otros ya existentes).

Esto se puede formalizar como sigue: dado un conjunto S del plano, para cada punto $x \in \mathbb{R}^2$ la *distancia de x a S* es:

$$d(x, S) := \text{mín}\{d(x, y) : y \in S\}.$$

Nuestro problema es encontrar un punto que maximiza la distancia a S. Si S es acotado, una solución obvia es tomar x "en el infinito" pero en la práctica necesitamos x en la misma región donde se encuentran los puntos de S. Podemos requerir, por ejemplo, que $x \in \text{conv}(S)$. El siguiente resultado nos dice que para encontrar ese x podemos usar el diagrama de Voronoi:

Teorema 3.4.2. *Dado un conjunto S de puntos en el plano, el punto de* $\text{conv}(S)$ *que maximiza la distancia a S es o bien un vértice del diagrama de Voronoi, o bien la intersección del borde de* $\text{conv}(S)$ *con una arista del diagrama de Voronoi.*

Demostración. Sea x un punto en $\text{conv}(S)$ que no es vértice del diagrama, tenemos que demostrar que x no maximiza la distancia, o lo que es lo mismo, queremos encontrar otros puntos cuya distancia a S es mayor que de x a S. Distinguimos varios casos:

- Si x está en el interior de $\text{conv}(S)$ y de una región de Voronoi $\text{Vor}(p)$, entonces p es el único punto de S con $d(x, S) = d(x, p)$. Podemos aumentar $d(x, S)$ moviendo x ligeramente en la dirección contraria al punto p a una posición x' con $d(x', S) = d(x', p) > d(x, p) = d(x, S)$.

- Si x está en el interior de $\text{conv}(S)$ y en el borde entre dos regiones de Voronoi $\text{Vor}(p)$ y $\text{Vor}(q)$, entonces p y q son los únicos punto de S con $d(x, S) = d(x, p) = d(x, q)$ (si hubiera un tercer punto r, entonces x sería el vértice común de tres regiones de Voronoi). Entonces podemos mover x ligeramente a lo largo del bisector de p y q y en dirección contraria al punto medio $(p + q)/2$, convirtiéndolo en un x' con

  $$d(x', S) = d(x', p) > d(x, p) = d(x, S).$$

  (Si x ya era el punto medio de p y q entonces podemos moverlo en cualquiera de las dos direcciones).

- Si x está en el borde de $\text{conv}(S)$ y en el interior de una región $\text{Vor}(p)$, x no es un vértice de $\text{conv}(S)$, así que está en el interior de una arista de $\text{conv}(S)$. Sea r la recta que contiene a esa arista. Podemos mover x a lo largo de r en dirección contraria al punto p' de r más cercano a p (p' es la intersección de r y la perpendicular a r por p). Igual que antes x se convierte en un x' de con $d(x', S) > d(x, S)$.

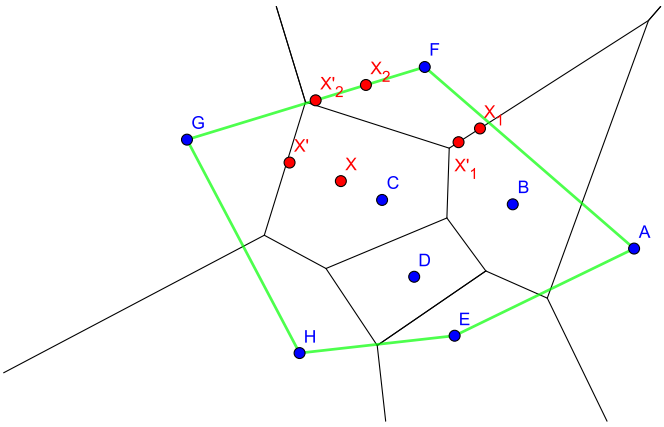

Figura 3.3: Explicación gráfica de la demostración del Teorema 3.4.2

□

3.4.3. Cada región de Voronoi es una región poligonal convexa, acotada o no

Sea p_i un punto de S. Para cada $p_j \in S \setminus \{p_i\}$ tenemos que la región

$$\{x : \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j)\}$$

es un semiplano cerrado, delimitado por el bisector de p_i y p_j. La región de p_i es la intersección de esos $n-1$ semiplanos. Lo semiplanos son convexos y toda intersección de convexos es convexa.

3.4.4. Cada vecino más cercano de p_i define una arista del polígono de Voronoi $V(i)$

Sea p_j el vecino más cercano de p_i y sea v el punto medio del segmento $p_i p_j$. Supongamos que v no está en la frontera de $V(i)$. Entonces, el segmento de línea $\overline{p_i v}$ cruza algún borde de $V(i)$, digamos la bisectriz de $\overline{p_i p_k}$ en u (Figura 3.4). Entonces $d(p_i, u) < d(\overline{p_i v})$, así que $d(\overline{p_i p_k}) \leq 2d(\overline{p_i u}) < 2d(\overline{p_i v}) = d(\overline{p_i p_k})$ y tendremos p_k más cerca de p_i que p_j, lo cual es contradictorio.

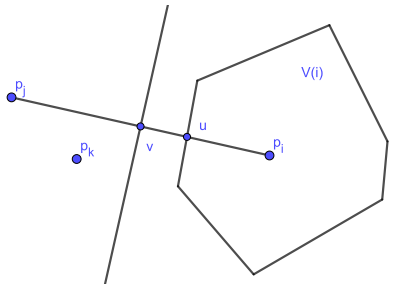

Figura 3.4: Cada vecino más cercano de p_i define una arista del polígono Voronoi $V(i)$

3.4.5. Un polígono $V(i)$ es no acotado si y solo si el punto p_i está en el borde del casco convexo de S.

Si p_i no está en el borde del casco convexo de S, entonces es interno a algún triángulo $p_1p_2p_3$ con vértices en el borde de conv(S) (para ver esto basta con triangular conv(S)).

Consideremos los círculos C_{12}, C_{13} y C_{23} determinados por p_i y cada uno de los tres pares de vértices $\{p_1, p_2\}$,$\{p_1, p_3\}$ y $\{p_2, p_3\}$, respectivamente. (Veáse la Figura 3.5 para toda esta explicación). Cada uno de estos círculos tiene un radio finito, para simplificar la demostración, suponemos que los puntos están "en posición general", o lo que es lo mismo, no hay tres puntos alineados. En el círculo C_{12} el arco externo A_{12} es el arco circular entre p_1 y p_2 que no contiene p_i; Es sencillo mostrar que cualquier punto de A_{12} está más cerca de p_1 o de p_2 que de p_i. Sea C un círculo que encierre C_{12}, C_{13} y C_{23}. Afirmamos que cualquier punto x que está fuera de C está más cerca de uno de p_1, p_2 o p_3 que p_i. En efecto, consideremos el segmento $\overline{xp_i}$. Por el teorema de la curva de Jordan [1], $\overline{xp_i}$ interseca uno de los lados o triángulo $p_1p_2p_3$, digamos $\overline{p_1p_2}$; por lo tanto interseca también A_{12} en el punto u. Pero u está más cerca de p_1 o p_2 que de p_i, de ahí la afirmación. Puesto que x está más cerca de p_1, p_2 o p_3 que de p_i, $V(i)$ está contenido enteramente dentro de C y por lo tanto está limitado. Por el contrario, supongamos que $V(i)$ está limitada y que $e_1, e_2, ..., e_k (k \geq 3)$ sea la secuencia de sus aristas límite. Cada e_h pertenece a la bisectriz de un segmento $\overline{p_ip_h}, p_h \epsilon S$. Es inmediato concluir que p_i es interno al polígono $p_1p_2, ..., p_k$ es decir, p_i no está en el casco convexo de S. Dado que sólo los polígonos no acotados pueden tener rayos como bordes, los rayos del diagrama de Voronoi corresponden a pares de puntos adyacentes de S en el casco convexo.

[1] Teorema de la Curva de Jordan: Toda curva cerrada simple del plano divide al plano en dos componentes conexas disjuntas que tienen a la curva como frontera común. Una de estas componentes está acotada (el interior de la curva) y la otra es no acotada y se le llama exterior.

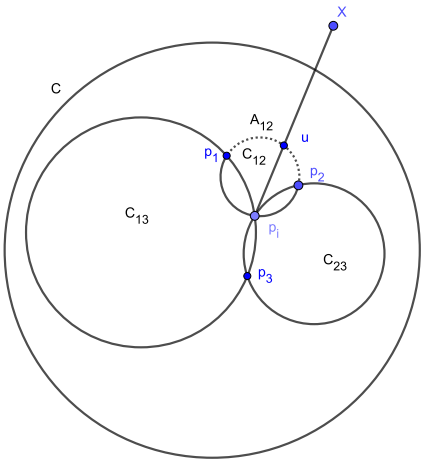

Figura 3.5: Ayuda a visualizar la demostracion de la propiedad 3.4.5

3.5. Aplicaciones de los diagramas de Voronoi.

En esta sección vamos a ver que una vez calculado el diagrama de Voronoi hay varios problemas que se pueden resolver de manera fácil con la información que nos da el diagrama.

3.5.1. Vecinos más cercanos

Dado un conjunto S formado por n puntos en el plano, queremos encontrar para cada b de S el punto a de S más cercano a b. Usaremos la notación $a \to b$ para indicar eso. Es decir:

$$a \to b \qquad \Leftrightarrow \qquad d(a,b) = \min_{c \in S} d(a,c).$$

La solución al problema es un conjunto de pares ordenados (a, b), donde $a \to b$. Podemos representar esta relación como un grafo dirigido. Véase el ejemplo de la figura 3.6.

Notemos que la relación no es necesariamente simétrica, es decir, $a \to b$ no implica necesariamente $b \to a$. Hay que tener en cuenta también que cada punto no es el vecino más cercano de un único punto (es decir, $\to$ no es necesariamente una función). Un par que sí satisfaga la simetría ($a \to b$ y $b \to a$) se llama un *par de reciprocidad*.

El problema de todos los vecinos más cercanos es transformable en tiempo lineal al diagrama de Voronoi y así puede resolverse en tiempo $O(nlogn)$, el cual es optimo.

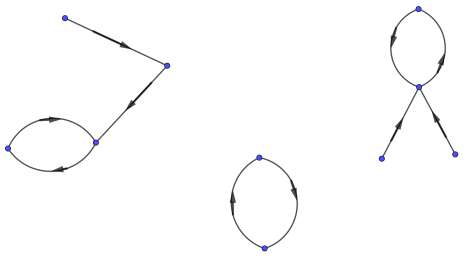

Figura 3.6: La relación de vecino más cercano en un conjunto de puntos

Cada vecino más cercano de un punto p_i, define una arista de $V(i)$. Para encontrar un vecino más cercano de p_i, solo es necesario explorar cada arista de $V(i)$. Como cada arista pertenece a dos polígonos de Voronoi, ninguna arista se examinara más de dos veces. Por lo tanto, dado el diagrama de Voronoi, todos los vecinos más cercanos se pueden encontrar en tiempo lineal.

Obviamente, dado que el par más cercano se transforma en tiempo lineal para todos los vecinos más cercanos, entonces el diagrama de Voronoi también se puede usar de manera óptima para resolver el problema.

3.5.2. Árbol generador Euclídeo mínimo: (Euclidean minimum spanning tree)

Dados n puntos en el plano, construir un árbol de longitud total mínima cuyos vértices son los puntos dados.

Una solución a este problema, significa dar una lista de los $n-1$ pares de puntos que comprenden las aristas del árbol. Un árbol de este tipo se muestra en la Figura 3.7.

El problema del árbol euclidiano de expansión mínima (EMST) es un componente común en aplicaciones que implican las redes. Si se desea establecer un sistema de comunicaciones entre los nodos de interconexión que requieren n cables, utilizando el EMST dará lugar a una red de coste mínimo.

Para entender mejor el árbol generador Eucludio mínimo tenemos los siguientes lemas.

Lema 3.5.1. *Sea $G = (V, E)$ una gráfica con aristas ponderadas y que V_1, V_2 sean una partición del conjunto V. Hay un árbol de expansión mínima de G que contiene el más corto entre las aristas con un extremo en V_1 y otro en V_2*

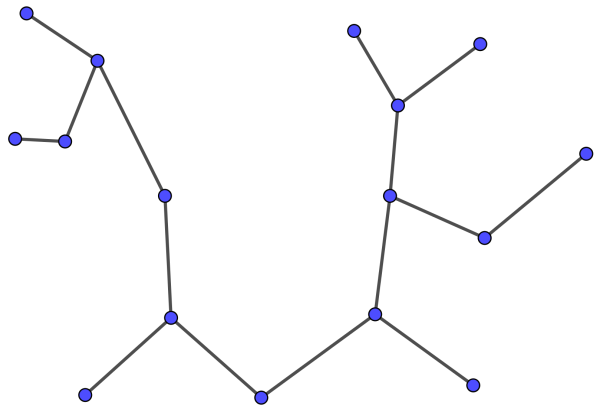

Figura 3.7: Un mínimo árbol generador en un conjunto de puntos en el plano

Demostración. En nuestro caso, el conjunto de vértices V es el conjunto de puntos de S, y la longitud de una arista es la distancia euclidiana entre sus dos puntos finales. El algoritmo EMST maneja en cada paso, un bosque de árboles.El bosque inicial es la colección de los puntos. El paso general del algoritmo se ejecuta de la siguiente manera:

1. Seleccionar un árbol T del bosque.
2. Encontrar una arista (u', v'), de modo que $d(u', v') = \text{mín}\{d(u, v) : u \in T, v \in S - T\}$.
3. Si T' es el árbol que contiene a v', unir T y T' uniéndolas por las aristas (u', v').

El algoritmo finaliza cuando el bosque está formado por un solo árbol. □

Lema 3.5.2. *Sea S un conjunto de puntos en el plano, y $\triangle\ (p)$ denota el conjunto de puntos adyacentes de $p \in S$ en la triangulación de Delemay de S. Para cualquier partición S_1, S_2 de S, si $\overline{qp}$ es el segmento más corto entre los puntos S_1 y los puntos de S_2, entonces q pertenece a $\triangle\ (p)$.*

Demostración. Sea $\overline{pq}$ la mínima distancia entre los puntos de S_1 y de S_2, con $p \in S_1$ y $q \in S_2$. Se puede afirmar que $q \in \triangle\ (p)$. Pero, si $q \notin \triangle\ (p)$, el segmento de la bisectriz perpendicular $\overline{pq}$ no contiene un segmento de límite $V(p)$. Esto implica que $V(p)$ interseca $\overline{pq}$ en un punto u entre p y el punto medio M de $\overline{pq}$. La arista de Voronoi l que contiene u es la bisectriz perpendicular de un segmento $\overline{pp'}$, donde $p' \in S$. Para todas las opciones posibles de u y de l, p' está en el interior del disco C con centro M y diámetro $\overline{qp}$. Ahora tenemos dos casos:

1. $p' \in S_1$. En este caso $d(q, p') < d(q, p)$ donde p', y no p, esta más cerca de q. Contradicción.

2. $p' \in S_2$. En este caso $d(p, p') < d(q, p)$ donde p', y no q, esta más cerca de p. Contradicción.

□

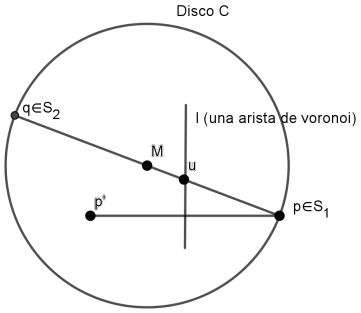

Figura 3.8: Ilustracion de la prueba $q \in \triangle\ (p)$

3.5.3. Búsqueda del vecino más cercano (Nearest-neighbor search)

Dados n puntos en el plano cómo de rápido se puede encontrar un vecino más cercano de cada nuevo punto de consulta q. Es trivial resolver este problema en tiempo $O(dn)$ en d dimensiones, simplemente calculando la distancia de q a los n puntos originales. Pero estamos interesados en usar un pre-procesamiento de los puntos dados para acelerar las búsquedas (suponemos que los n puntos no cambian, y que queremos resolver el problema para muchas instancias de q).

La manera de hacerlo es como sigue: primero dividimos el plano en franjas verticales usando para ello una recta vertical por cada vértice del diagrama de Voronoi. El número de vértices está acotado por $O(n)$. Luego, en cada franja vertical, calculamos las ecuaciones de las rectas que separan las distintas regiones, de modo que cada franja queda dividida en (a lo sumo) n trapecios (véase la Figura 3.9). Ahora, dado un punto $\mathbf{p} = (x, y) \in \mathbb{R}^2$, para averiguar en qué región de Voronoi está hacemos lo siguiente:

1. Por búsqueda binaria en la coordenada x averiguamos en qué franja vertical está $\mathbf{p}$, en tiempo $O(\log n)$.

2. Una vez que tenemos eso, por búsqueda binaria en las ecuaciones de las rectas en dicha franja, averiguamos en qué trapecio está $\mathbf{p}$, en tiempo también $O(\log n)$.

Por tanto, la búsqueda del vecino más cercanos se realizar en el tiempo $O(\log n)$, usando el almacenamiento $O(n)$ y el tiempo de pre-procesamiento $O(n \log n)$, lo cual es óptimo. Este $O(n \log n)$ es el tiempo que se usa para construir el diagrama Voronoi de los puntos.

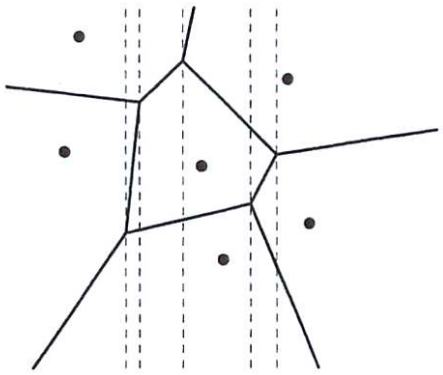

Figura 3.9: Preprocesado de la nube de puntos para resolver búsquedas de punto más cercano. (Figura tomada de [8]).

Hay una multitud de aplicaciones (Figura 3.10) para que tal búsqueda sea tan rápida, posiblemente el más importante es el problema de clasificación. Un método de clasificación es la regla del vecino más cercano, que establece que cuando un objeto debe clasificarse como perteneciente a una serie de poblaciones conocidas, se debe colocar en la población correspondiente a su vecino más cercano.

Una aplicación similar ocurre en la recuperación de información, donde se recupera el registro que mejor coincide con el registro de la consulta. Si muchos objetos se van a procesar, ya sea en tareas de clasificación o en tareas de recuperación, debemos ser capaces de realizar la búsqueda del vecino más cercano rápidamente.

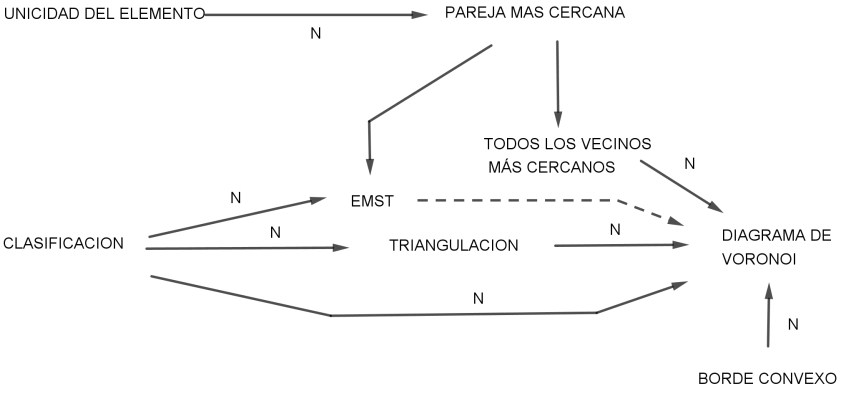

Figura 3.10: Relación entre prototipos computacionales y problemas de proximidad

3.6. Diagrama de Voronoi de puntos más lejanos

Vimos en la Sección 3.5 que el punto que "maximiza la distancia mínima a los puntos de S" es un vértice del diagrama de Voronoi. Supongamos que estamos interesados en lo contrario: queremos "minimizar la distancia máxima a S". Por ejemplo, queremos colocar un servicio o un negocio en el que los puntos de S representan los usuarios o los clientes, y queremos que ninguno de ellos esté demasiado lejos del nuevo servicio.

La solución nos la da el diagrama de Voronoi de puntos más lejanos.

Definición 3.6.1. Dado un conjunto de puntos en el plano, $P = \{p_1, ..., p_n\} \subset \mathbb{R}^2$, el *diagrama de Voronoi del punto más lejano de P* es la descomposición del plano en regiones asociadas a los puntos, siendo la región i el lugar geométrico de los puntos del plano cuya distancia a p_i es mayor que la distancia a cualquiera de los sitios restantes.

El diagrama de Voronoi de puntos más lejanos tiene las siguientes propiedades:

1. El diagrama de Voronoi de punto más alejado divide el plano en regiones poligonales convexas.

   Demostración. Sea p_i un punto de S. Para cada $p_j \in S \setminus \{p_i\}$ tenemos que la región
   $$\{x \in \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j)\}$$
   es un semiplano cerrado, delimitado por el bisector de p_i y p_j. La región de p_i es la intersección de esos $n-1$ semiplanos. Lo semiplanos son convexos y toda intersección de convexos es convexa. □

2. La región de Voronoi de punto más lejano de p_i es no vacía si, y solo si, p_i es un vértice de conv(S). Las regiones son todas no acotadas.

   Demostración. Primero veamos que si la región de p_i es no vacía entonces p_i es un vértice de conv(S). Sea $x \in \mathbb{R}^2$ en Vorlej(p_i). Entonces $d(x, p_i) =$ máx$\{d(x, q) : q \in S\}$. Sea C la circunferencia que pasa por p y con centro en x. Entonces $p \in C$ y $S \subseteq \text{conv}(C) \Rightarrow \text{conv}(S) \subseteq \text{conv}(C)$. Como p_i está en la circunferencia, conv$(S) \subseteq$ conv(C) implica que p_i es un vértice de conv(C).

   Para el recíproco, supongamos que p_i es un vértice de conv(S). Entonces, existe una recta r que corta a conv(S) solo en el punto p_i. Sea l la semirrecta perpendicular a r que parte por p_i en el lado de conv(S). Sea x un punto de l. A medida que movemos x hacia el infinito, la circunferencia con centro en x y que pasa por p_i tiende a la recta r. Por tanto, para cualquier x suficientemente lejano, todos los puntos de $S \setminus \{p_i\}$ están dentro de la circunferencia. Eso implica que Vorlej(p_i) es no vacía y es no acotada en la dirección de l. □

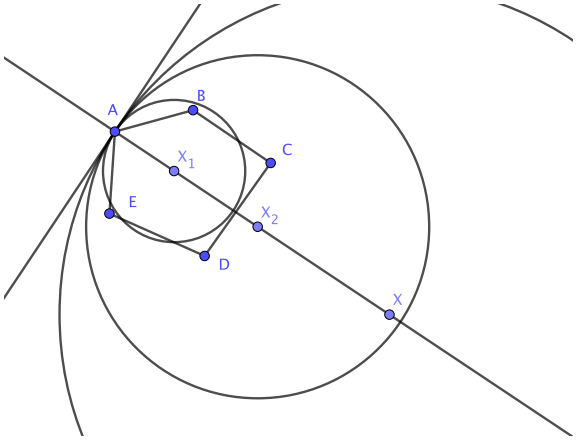

Figura 3.11: Ilustración de la demostración

3. Todo vértice del diagrama es el centro de una circunferencia C que pasa por al menos tres de los puntos y contiene a todos los demás o dentro o sobre ella.

   Demostración. Sea S el conjunto de puntos del diagrama de Voronoi. Sea v un vértice del diagrama, es decir, v está en al menos tres regiones Vorlej(p_i), Vorlej(p_j), Vorlej(p_k). Eso quiere decir que $d(v, p_i) = d(v, p_j) = d(v, p_k)$ y esa distancia es mayor o igual que $d(v, q)$ para todo q nS. Es decir, la circunferencia con centro en v y radio $d(v, p_i)$ pasa por p_i, p_j y p_k y todos los demás están o dentro o sobre ella. □

La propiedad más importante de este diagrama es:

Teorema 3.6.2. *Dado un conjunto S de puntos en el plano, el punto de $\mathbb{R}^2$ que minimiza la máxima distancia a los puntos de S es o bien un vértice del diagrama de Voronoi de puntos más lejanos o bien el punto medio $(p+q)/2$ de dos puntos p y q cuyas regiones* Vorlej(p) *y* Vorlej(q) *son adyacentes..*

Demostración. Sea x un punto en conv(S) que no es vértice del diagrama, tenemos que demostrar que x no minimiza la máxima distancia, o lo que es lo mismo, queremos encontrar otros puntos cuya máxima distancia a S es menor que de x a S. Distinguimos dos casos:

- Si x está en el interior de una región Vorlej(p) del Voronoi de puntos más lejanos, entonces p es el único punto de S con $\max_{q \in S} d(x, q) = d(x, p)$. Podemos disminuir la distancia máxima moviendo x ligeramente en la dirección hacia p a una posición x' con $\max_{q \in S} d(x', q) = d(x', p) < d(x, p) = \max_{q \in S} d(x, q)$.

- Si x está en el borde entre dos regiones Vorlej(p) y Vorlej(q) pero $x \neq (p+q)/2$ y x no es un vértice del Voronoi, entonces movemos x a lo largo del bisector de p y q acercándolo al punto medio de $(p+q)/2$. Eso disminuye la distancia a p y a q, que eran los únicos puntos que maximizaban la distancia a x.

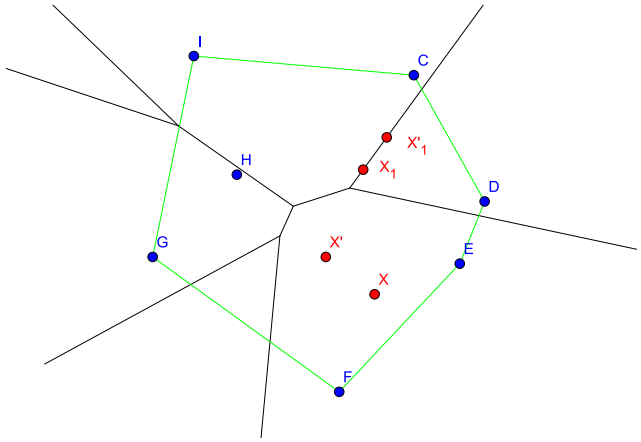

Figura 3.12: Explicación gráfica de la demostración del Teorema 3.6.2

□

Observación 3.6.3. El diagrama de Voronoi usual y el de puntos más lejanos son casos particulares de lo siguiente. Dado $P = \{p_1, ..., p_n\} \subset \mathbb{R}^2$, el *diagrama de Voronoi de orden k* es la descomposición del plano en regiones asociadas a las k-tuplas de puntos. Siendo la región de $\{i_1, \ldots, i_k\}$ es el lugar geométrico de los puntos del plano cuyos k puntos de S más cercanos son precisamente esos k.

En este lenguaje, el diagrama de Voronoi usual y el de puntos más lejanos son los diagramas de Voronoi de orden 1 y $n-1$, respectivamente.

Capítulo 4

Ejemplos

A continuación vamos a mostrar una serie de ejemplos de resolución del problema de Weber y de construcción del Diagrama de Voronoi. Para el problema de Weber hemos implementado un algoritmo utilizando el software Sagemath [15], un sistema algorítmico computacional que destaca por estar construido sobre varios paquetes matemáticos como NumPy, Sumpy o Maxima. Es un software libre y usa el lenguaje python.

Para el Diagrama de Voronoi se ha utilizado Geogebra, ya que dicho programa tiene un comando para la construcción directa de diagramas de Voronoi. Geogebra es a la vez un procesador geométrico y un procesador algebraico; es decir, un software que reúne geometría, álgebra, estadística y cálculo.

4.1. Algoritmo de Weiszfeld

El primer ejemplo es una implementación del algoritmo de Weiszfeld para resolver problemas de Weber en el plano, tal como se discutió en la Sección 2.4

```
# Algoritmo de Weisfeld
#   el  algoritmo toma como input tres listas a,b,w de la
#   misma longitud y dos numeros x0 e y0:
#       a = lista de coordenadas x de los puntos
#       b = lista de coordenadas y de los puntos
#       w = lista de pesos para los puntos
#       x0 e y0: coordenadas del punto tomado como solucion inicial
def Weiszfeld(a,b,w,x0,y0):
    diferencia=1
    n=len(a) #numero de puntos
    eps1=0.01 #tolerancia en relacion a la precision de ordenador
    while diferencia > eps1:
        A=0
        B=0
        C=0
```

```
        for i in [0..n-1]:
            d=sqrt((x0-a[i])^2+(y0-b[i])^2);
            if d==0:
                print("ERROR; la iteración nos lleva al punto", (x0,y0) ,
                    "que es uno de los puntos de demanda");
                return
            A=A+(w[i]*a[i])/d;
            B=B+(w[i]*b[i])/d;
            C=C+w[i]/d;
        x=A/C
        y=B/C
        diferencia = ((x-x0)^2+(y-y0)^2).n()
        x0=x.n();
        y0=y.n();
        print(x0,y0, diferencia)
    return x0,y0
```

Ejemplo de Khun

Como primer ejemplo, vamos a hacer un ejemplo originalmente de Khun, que aparece también en [5]. El problema tiene cuatro puntos de demanda, que son $A_1(20, 0)$, $A_2(59, 0)$, $A_3(-20, 48)$ y $A_4(-20, -48)$. Los puntos de demanda A_1 y A_2 tienen peso $w_1 = w_2 = 5$, mientras que los otros dos puntos de demanda tienen pesos $w_3 = w_4 = 13$.

Su solución óptima es $(0, 0)$ como se demuestra con el siguiente argumento: Por la simetría del problema, la solución se encuentra en el eje X. Por otro lado, tiene que estar en el intervalo $[-20, 20]$, porque en $[20, 59]$ las contribuciones de A_1 y A_2 se cancelan y las de A_3 y A_4 tiran hacia la izquierda. Planteando la cancelación de fuerzas se ve que el equilibrio está en el punto $(0, 0)$. Esto se obtiene también con el algoritmo tomando como punto inicial un punto más o menos aleatorio:

```
a=(20, 59, -20, -20)
b=(0, 0 ,-48 ,48)
w=(5 ,5 ,13, 13)
x0=18
y0=9
weiszfeld(a,b,w,x0,y0)

(8.53629379382984, 2.15689849508429, 136.389773363283)
(5.12530247252180, 0.747128040899221, 13.6223145275319)
(2.85127542485096, 0.309813885789692, 5.36244268379774)
(1.49774509245009, 0.142042830135851, 1.86019148784440)
(0.760667065895220, 0.0686846683313886, 0.548665437133351)
(0.379389298934346, 0.0341276232972508, 0.146566924940162)
(0.187472944712967, 0.0171892064404208, 0.0371187969832416)
```

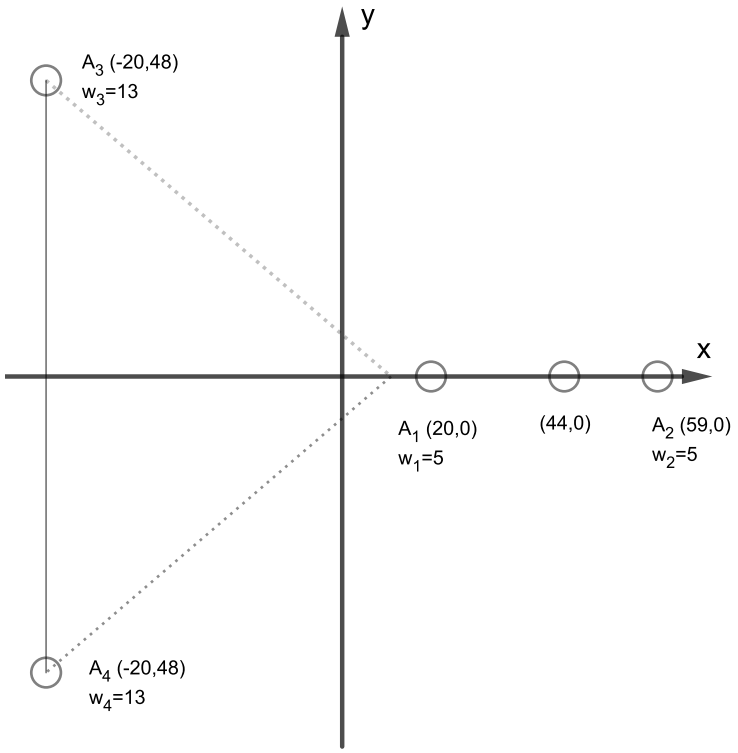

Figura 4.1: Datos del ejercicio de Khun

```
(0.0922080701221003, 0.00871618453286626, 0.00914718843105944)
(0.0922080701221003, 0.00871618453286626)
```

Sin embargo, tomando como punto inicial $(x_0, y_0) = (44, 0)$, en el primer paso el algoritmo de Weiszfeld nos lleva al punto de demanda A_1 y se atasca en ese punto, a pesar de que no es el óptimo.

```
a=(20, 59, -20, -20)
b=(0, 0 ,-48 ,48)
w=(5 ,5 ,13, 13)
x0=44
y0=0
weiszfeld(a,b,w,x0,y0)
(20.0000000000000, 0.000000000000000, 576.000000000000)
('ERROR; la iteracion nos lleva al punto', (20.0000000000000,
```

```
0.000000000000000), 'que es uno de
de los puntos de demanda'
```

Ejemplo de Drezner y Hamacher

El segundo ejemplo es un problema tomado de Drezner y Hamacher [6],también aparece en [5], el cual es extremadamente difícil de ejecutar con el algoritmo de Weiszfeld.

El problema consiste en cinco puntos de demanda, cuatro de ellos ubicados en las cuatro esquinas de un cuadrado unidad, $A_1 = (0, 0)$, $A_2 = (0, 1)$, $A_3 = (1, 0)$ y $A_4 = (1, 1)$ con pesos $w_1 = w_2 = w_3 = w_4 = 1$ y un quinto punto de demanda $A_5 = (100, 100)$ con peso $w_5 = 4$. Tomamos como punto inicial el punto $(x_0, y_0) = (50, 0)$.

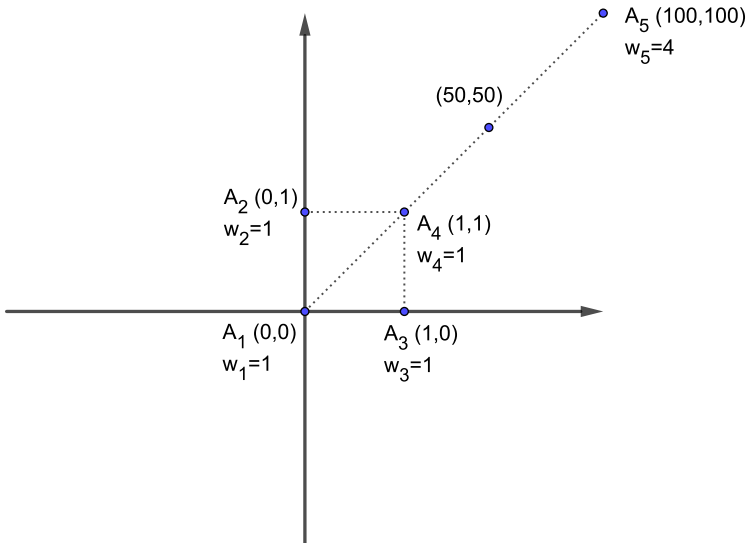

Figura 4.2: Datos del ejercicio de Drezner

Por simetría, la solución debe estar a lo largo de la diagonal $x = y$ y entre $[1, 1]$ y $[50, 50]$. Si se consideran los puntos en la diagonal cercanos a A_5, este último ejerce una fuerza de cuatro unidades hacia arriba por la diagonal. Los pesos en A_1 y A_4 ejercen una fuerza de una unidad hacia abajo por la diagonal. Cada uno de los otros puntos A_2 y A_3 ejercen una fuerza a lo largo de la linea que conectan con A_5, y su componente a lo largo de la diagonal es un poco más pequeña que la unidad. Por lo tanto, estemos donde estemos la resultante tira hacia A_5 y por tanto A_5 es el óptimo. Pero como la resultante tiende a cero a medida que nos acercamos a A_5 la convergencia es muy lenta y el algoritmo de

Weiszfeld requiere muchas iteraciones para llegar al punto óptimo, como se ve al correr el programa:

```
a=(0,0,1,1,100)
b=(0,1,0,1,100)
w=(1,1,1,1,4)
x0=50
y0=0
weiszfeld(a,b,w,x0,y0)
(31.0375748670167, 31.0340395796629, 1322.68517955608)
(31.0372257015596, 31.0372253732212, 0.0000102711971126933)

 (31.0372257015596, 31.0372253732212)
```

Ejemplo de Chen

Por último, tenemos el siguiente ejemplo tomado de [5]: cuatro puntos de demanda $A_1 = (0, -10)$, $A_2 = (0, 10)$, $A_3 = (-10, 0)$ y $A_4 = (10, 0)$. Vamos a dar pesos iguales a A_1 y A_2, $w_1 = w_2 = w$ y peso mayor a A_4 que a A_3, para que el óptimo esté a lo largo del eje x y en el segmento que va de $(0, 0)$ a $(10, 0)$.

Lo que queremos calcular es qué han de cumplir w_3 y w_4 para que el óptimo esté en A_4. Para ello sabemos cuales son las fuerzas que ejercen los puntos,$F_1 = (-\cos\alpha w, \operatorname{sen}\alpha w)$, $F_2 = (-\cos\alpha w, -\operatorname{sen}\alpha w)$, $F_3 = (-w_3, 0)$, $F_1 = (w_4, 0)$. El punto de equilibrio le encontramos cuando la suma de las fuerzas es igual a cero. En este caso sabemos que $\sum_1^4 F_i = (w_4 - w_3 - 2w\cos\alpha, 0)$ y como queremos que el equilibrio este cerca de A_4, el resultado que obtenemos es

$$w_4 \geq \sqrt{2}w + w_3.$$

```
a=(0, 0 , -10 , 10)
b=(-10, 10, 0, 0)
w=(5 ,5 , 0 , 5*sqrt(2))
x0=0
y0=0
weiszfeld(a,b,w,x0,y0)
(4.14213562373095, 0.000000000000000, 17.1572875253810)
(5.66454497350522, 0.000000000000000, 2.31773022828010)
(6.52110812168597, 0.000000000000000, 0.733700426821331)
(7.08161329224151, 0.000000000000000, 0.314166046219486)
(7.48044565780783, 0.000000000000000, 0.159067255823227)
(7.78014785787277, 0.000000000000000, 0.0898214087237651)
(8.01425619226788, 0.000000000000000, 0.0548067122332559)
(8.20252867664543, 0.000000000000000, 0.0354465283736932)
(8.35742398603266, 0.000000000000000, 0.0239925568701663)
(8.48721463845583, 0.000000000000000, 0.0168456134564316)
(8.59762221157287, 0.000000000000000, 0.0121898322015938)
(8.69273714664063, 0.000000000000000, 0.00904685087294568)
```

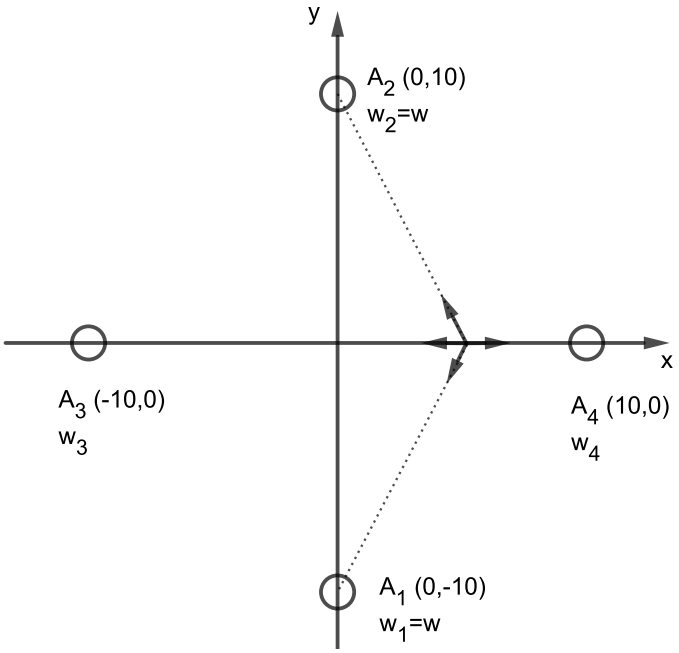

Figura 4.3: Datos del ejercicio de Chen

```
(8.69273714664063, 0.000000000000000)
```

Introduciendo los datos en el algoritmo podemos observar que el punto optimo que nos da es (8.69273714664063, 0.000000000000000), el cual es muy próximo a A_4, como se nos pedía en el problema.

4.2. Diagrama de Voronoi usual

Los diagramas de Voronoi los calculamos directamente con la función `Voronoi` de GeoGebra.

Aeropuertos de España

En el primer ejemplo,queremos encontrar el lugar idóneo para ubicar un aeropuerto en España, sin contar los aeropuertos que se encuentran en las islas

Canarias, ni Ceuta y Melilla. Para ello tenemos los datos de todos los correspondientes aeropuertos, aplicando a estos el algoritmo de Voronoi obtenemos el diagrama de la Figura 4.4.

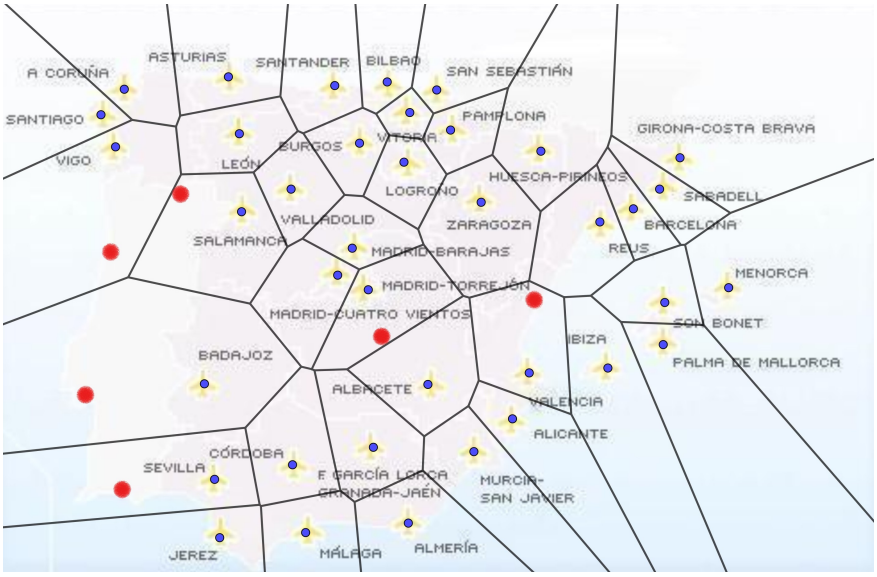

Figura 4.4: Diagrama de Voroni de los aeropuertos españoles

En primer lugar vamos a ver cuales son los aeropuertos mas cercanos a Santander, para ello nos ayudamos de la teoría vista anteriormente. En la Figura 4.5 observamos cuales son los aeropuertos mas cercanos a Santander. Aplicando el Lema 3.4.1, observamos que los aeropuertos que dan regiones de Voronoi vecinas a la de Santander son Asturias, León, Burgos, Valladolid y Bilbao, porque las circunferencias de la figura no tienen ningún aeropuerto en su interior, lo que nos indica que dichos aeropuertos son vecinos de Voronoi.

En cambio el aeropuerto de Vitoria, que es también cercano al de Santander, no es vecino en el diagrama de Voronoi porque, aplicando la teoría, vemos que en la circunferencia de tres puntos (Santander, Burgos y Vitoria) nos encontramos que dentro hay un punto (Bilbao). Ver Figura 4.6.

Figura 4.5: Cálculo de los vecinos del aeropuerto de Santander en el diagrama de Voroni de los aeropuertos españoles

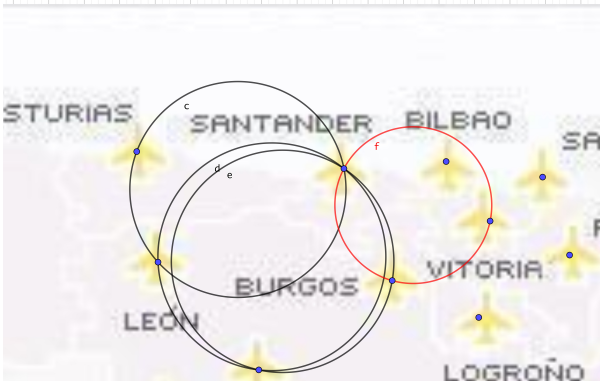

Figura 4.6: Cálculo de los vecinos del aeropuerto de Santander en el diagrama de Voroni de los aeropuertos españoles

Como ya hemos dicho, en la figura 4.4, se muestra el diagrama de voronoi completo. Los puntos rojos en esta figura indican los aeropuertos de Portugal y los aeropuertos de Ciudad Real y Castellón, que no se han tenido en cuenta para el cálculo del diagrama. En el caso del aeropuerto de Ciudad Real fue un aeropuerto privado y actualmente está cerrado e inactivo, dicho aeropuerto fue construido como alternativa del aeropuerto Madrid-Barajas. El aeropuerto de Castellón estuvo cerrado e inactivo hasta el 2015 donde empezó a operar como aeropuerto privado.

Podemos observar en la figura cómo el centro del país es uno de los lugares donde las regiones de Voronoi son mas grandes, ocurre lo mismo con la zona oeste del país, en estas regiones podrían ser uno de los sitios en los que podríamos

poner un aeropuerto nuevo, aunque en las regiones de Salamanca y Badajoz, deberíamos de tener en cuenta que en Portugal también hay aeropuertos (indicados como los puntos rojos como hemos mencionado anteriormente). También hay que tener en cuenta que aunque estas regiones de Voronoi son más grandes en extensiónse se da la circunstancia de que son lugares cuya población es escasa.

Otro de los detalles o curiosidades que se ven en la figura es que la mayoría de los aeropuertos se encuentran en zonas de costa.

Talleres mecánicos en Solares

En el segundo ejemplo, queremos encontrar el lugar idóneo para ubicar un taller mecánico en el municipio de Solares. Para ello tenemos los datos de todos los talleres ubicados en dicho municipio. Aplicando a estos el algoritmo de Voronoi obtenemos el diagrama de la figura 4.7.

Como se puede observar, dichos talleres están bastante bien ubicados, ya que todos abarcan grandes regiones de población. Los dos puntos negros marcados en la figura son las ubicaciones donde se podría colocar un nuevo taller; estos dos puntos pueden ser los lugares óptimos ya que son las dos regiones con mayor población y por lo tanto podría haber un gran número de potenciales clientes.

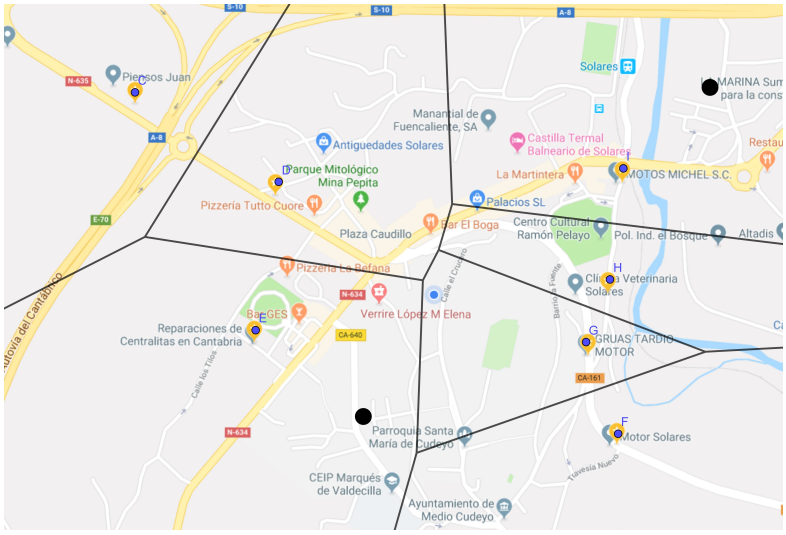

Figura 4.7: Diagrama de Voronoi de los talleres de Solares, con los dos posibles talleres nuevos

Bibliografía

[1] M. de Berg, M. van Kreveld, M. Overmars y O. Schwarzkopf, *Computtional geometry: algorithms and applications*, Springer, 1997.

[2] Binay Bhattacharya, "Extensions of Voronoi Diagrams", apuntes del curso "Computational Geometry", 2011, Simon Fraser University. `http://www.cs.sfu.ca/~binay/813.2011/ExtensionsVD.pdf`

[3] Norman Biggs, *Algebraic graph theory*, Cambridge : Cambridge University Press, 2001.

[4] J. Brimberg, The Fermat-Weber location problem revisited, *Mathematical Programming*, Volume 71, pages 71–76, Springer, 1995.

[5] Reuven Chen, Noniterative Solution of Some Fermat-Weber Location Problems, *Advances in Operations Research*, Volume 2011, Article ID 379505, 10 pages, Hindawi Publishing Corporation doi:10.1155/2011/379505

[6] Zvi Drezner y Horst W. Hamacher, *Facility Location Applications and Theory*, Springer, 2002.

[7] Harold M. Edwards, *Fermat's last theorem : a genetic introduction to algebraic number theory*, Springer, 1977.

[8] M. Joswig y T. Theobald, Polyhedral and Algebraic Methods in Computational Geometry, Springer-Verlag, 2013.

[9] Harold W. Kuhn, A note on Fermat's problem, *Mathematical Programming*, Volume 4, pages 98-107, Springer, 1973.

[10] Robert F. Love, James G.Morris y George O.Wesolowsky, *Facilities Location: Models and Methods*, North-Holland, 1988.

[11] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, *Spatial tessellations : concepts and applications of Voronoi diagrams*, Chichester, 1992.

[12] Franco P. Preparata y Michael Ian Shamos, *Computational Geometry An Introduction*, Springer, 1985.

[13] Dolores R. Santos Peñate, "Problemas de Localización", presentación como parte del curso "Métodos matemáticos en ciencias sociales, economía, finanzas y administración de empresas", SCTM 2005, Univerisdades de La Laguna y de las Palmas de Gran Canaria, https://imarrero.webs.ull.es/sctm05/modulo1lp/6/dsantosSlides.pdf

[14] Google Maps, https://www.google.com/maps/

[15] SageMath, http://www.sagemath.org

[16] Wikipedia, "Fermat point" https://en.wikipedia.org/wiki/Fermat_point